Asian Journal of Information Technology 5 (12): 1356-1360, 2006
© Medwell Journals, 2006

Semi-Formal and Formal Notation Automated Assessment

[1]Noraida Haji Ali, [2]Zarina Shukur and [2]Sufian Idris
[1]Department of Computer Science, Faculty of Science and Technology,
Kolej Universiti Sains dan Teknologi Malaysia, 21030 Kuala Terengganu,Terengganu, Malaysia
[2]Department of Computer Science, Faculty of Information Science and Technology,
Universiti Kebangsaan Malaysia, 46300 Bangi, Selangor, Malaysia

Abstract: Computer-Aided Software Engineering (CASE) is the application of information technology to system development activities, techniques and methodologies. CASE tools are software that automate or support one or more phases of a systems development life cycle. Basically students or academicians use CASE tools to understand the theory of the concept. However, some versions of these tools do not focus on the needs of students or academic users who will need more assistance to understand the theory itself. In this paper, we discussed the formalizing object-oriented and previous researches. Also we discussed the architecture of Object-Oriented Model Assessor (OOMA) which is proposed for development. The specification approach presented in this prototype is focused on the development of diagram assessment system for mapping diagram from UML model into Object-Z specification. The objectives of this research is to improve student's understanding on how to represent the system requirement in UML model and Object-Z specification model. In addition, it could also improve students's understanding on the relationship between these two models.

Key words: Object-oriented modeling, formal modeling, UML and object-Z

INTRODUCTION

Models are crucial in engineering principal because engineers to use them to describe shapes or actions of the construction they want to build. There are various kinds of models in programming contexts and one of them is object-oriented model. Object-oriented model can be illustrated by using semi-formal and formal notation. UML[1] is one of the semi-formal notation objects, while Object-Z[1] is formal. Although semi-formal notation like UML is popular and easy to comprehend by the users, it is vital for developers to know the formal method because it can assurance the creation of programming that fulfill the users' need[2]. Hence, it is an advantage for developers to master both models because it equip each other.

Model UML and Object-Z can be produced with the help of CASE tools like Rational Rose[3] for UML and Wizard for Object-Z[4]. The purpose of CASE tools is to reduce the gap between theory and practical. However, the tools do not focus on the learning environments which need more aids to increase users' understanding especially for students. Hoggarth and Lockyer[5] found that their students for System Analysis and Design course gave comments and responses that they understand the theory taught in class but it was hard to apply in practical form especially in given exercises or tasks. Furthermore, the available tools are unable to help the students to recognize the relationship between UML and Object-Z. To solve the above problems, this research suggested a design tool that can automatically evaluate the notation in model UML and Object-Z. We called the tool as Object-Oriented Model Assessor (OOMA). This tool is hoped to help students to understand how to represents a system need in semi-formal model like UML and formal model like Object-Z.

Related work: The previous researches show that object-oriented in semi-formal model can be used to produce object-oriented in formal model. On the whole, there are many researches on the formalization of both models which was done by the researchers for the past few years. Every research has their own differences and strengths in their respective approaches. Figure 1 shows the relationship between the researchers and the research in this formalization.

The early research started in 1990s when Object-oriented notation, OMT was used by the programming engineer to model the programming. In 1992, the formlization of model OMT and VDM++ were done by

Corresponding Author: Noraida Haji Ali,Department of Computer Science, Faculty of Science and Technology, Kolej Universiti Sains dan Teknologi Malaysia, 21030 Kuala Terengganu,Terengganu, Malaysia

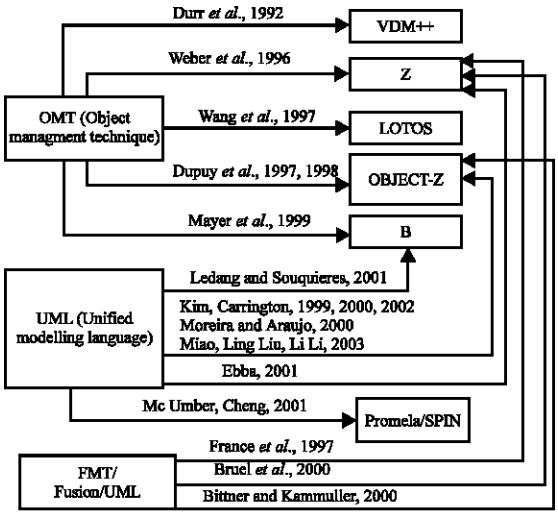

Fig. 1: The relationship between researchers and formalization studies

Durr *et al.*[6]. It was followed by other researchers who formalize the same object-oriented notation with other formal specification like Z, LOTOS, Object-Z and B. Wang[7] had formalized OMT model by using LOTOS while Weber[8] had combined class diagram and situation diagram (*statechart*). Meyer *et al.*[9] and Bertino *et al.*[10] had interpreted OMT diagram to B specification. Besides that, Dupuy *et al.*[11,12] had also formalized OMT by using Object-Z

Beginning in 1997, when object-oriented model, FMT/Fusion UML was produced, formalization was done to this model with the specification of Z formal and Object-Z. Formalization studies continued with the formalization of Object-Oriented FMT/Fusion UML with the formal Z specification and Object-Z. France *et al.* had developed a prototype of FuZE which automatically generated Z specification from Fusion Object Model[13]. Other studies had transformed FMT (fusion Modeling Techniques) to the formal Z specification with *FuZed* tools[14]. The Object-Oriented Model was updated to produce UML model which became the standardize model in Object-Oriented design. In late 1990s, formalization studies focused on UML model and Object-Z specification. The early study was done in 1999 by Kim and Carrington and later was developed by other researchers as was shown in Fig. 1. Some of the studies had developed tools for formalization purpose such as OZRose and RoZ. The tools are capable of automatically mapping the Object-Oriented to formal specification.

Object-oriented formalization: This section presents a brief description of the formalization of object-oriented

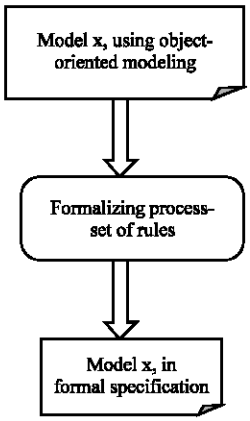

Fig. 2: Formalized object-orinted method with formale method

model. Formalizing object-oriented concepts in which formal specification techniques are used primarily to gain insights to the semantics of object-oriented notations. For the purpose of this paper, the example of formalizing object-oriented is given in this section. To get know how this process happened, we describe the translation of UML models of Object-Z specifications which proposed by Kim and Carrington[15]. A class in Object-Z is a template for objects that have a common state and operations. UML has been accepted as a standard OO modeling notation by OMG and is already popular in industry. Object-Z is an object-oriented extension to Z. designed specifically to facilitate specification in an object-oriented style. Using a well-defined object-oriented formal specification like Object-Z (where most of the fundamental concepts in object-orientation such as object-identity, class, inheritance and polymorphism are supported but the specification technique itself) to specify the UML :

- Gives a rigorous way of exploring the concepts embodied within the UML
- Make formal reasoning of UML models possible
- Provide a precise basis for mapping between different specifications language

According to previous researchs, there have several of terms to describe the formalizing process such as combining, integrating, translation, interpreting, transformation and mapping. However, these techniques have a same propose, that is, to formalized object-oriented method with formal method. Basically, this formalizing process can be indicated in Fig. 2.

A group of researchers have debated the benefits of incorporating formal modeling into object-oriented models[4]. Some of the advantages include it could:

- Help system developers have a better and clearer understanding on the flow of the systen and in particular the requirement specification.
- Enable components to be reused.
- Improve the analysis and design techniques in the development of software with a more coherent models.
- Be used in validating the semantic of the models.
- Make the analysis and design phases more accurate and complete.

Figure 3 show the example of UML class diagram for simple process in bank system. In this UML class diagram, there have a few notations such as classes, association relationship, generalization and so on. To get the scheme on how Object-Z specification was produced from UML class diagram figure, Fig. 3 shows the UML class diagram[15].

The diagram represents most UML class construct, namely class, association, composition association, association class and generalization. The diagram consists of two major entities in the system : *Customer* and *Account*. Each class has its own attributes and operations. Class *Account* is further classified into CheckingAccount using generalization in UML (represented by triangle symbol). The *CheckingAccount* is associated with class *checkbook*. The multiplicity constrains 1..20 means that an instance of *CheckingAccount* maps to at least one instance of class *checkbook* and at most twenty instances of *checkbook*. The *checkbook* has a composition relationship with class *check* (represented by filled diamond symbol). An association class transaction represents a relationship between class *customer* and *account* and has its own attributes.

UML class diagram was mapped to Object-Z specification with a set of rule which was laid down by Kim and Carrington[16]. Individual class constructs: classes, associations, association classes and generalization that appear in a class diagram map to their corresponding Object-Z constructs using the predefined mapping functions. To comprehend the mapping process, an *Account* class was taken as an example and Fig. 4 below shows the production of Object-Z specification according to the class diagram figure for *Account* class.

For *Account* class, it was mapped as a scheme which will be called as *Account* in Object-Z specification and for every attribution, which are *acNo* and *balance;* it was mapped as situation variable. While for its operations, which are *Withdraw* and *Deposit,* it was mapped as an operation schema in Object-Z specification.

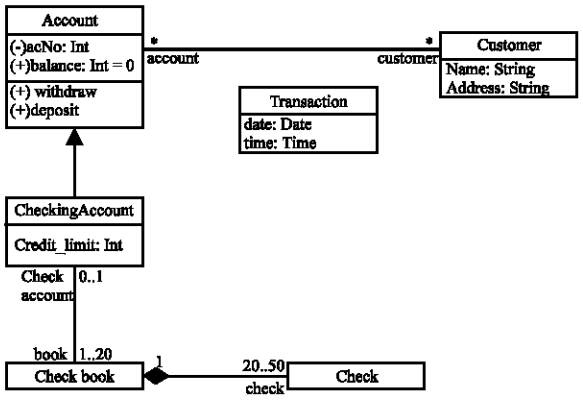

Fig. 3: UML class diagram for bank system

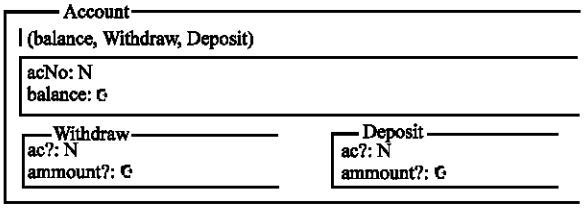

Fig. 4: Object-Z specification

Object oriented model assessor (OOMA) arhitecture: The proposed architecture of OOMA is shown as in Fig. 5.

Generally, OOMA consists of two modules; a module for teacher and a module for students.

Teacher module: This module requires inputs from teacher. Teacher can input UML class diagram as the model solution for given problem from Rational Rose Tool. The main process in this module is the mapping process. The function of this process is to map the UML class diagram into Object-Z specification. The generated Object-Z specification will be checked by a typechecker such as Wizard. The teacher is required to complete this generated Object-Z specification by input the predicates of the specification.

Student module: This module requires two inputs from the students; UML class diagram and Object-Z specification. UML class diagram will be drawn by using Rational Rose. The system will check the class diagram based on three aspects;

- Syntax Analysis
  This process will check the syntax for UML class diagram. For example, necessity items in an object like class name, attributes and operation.

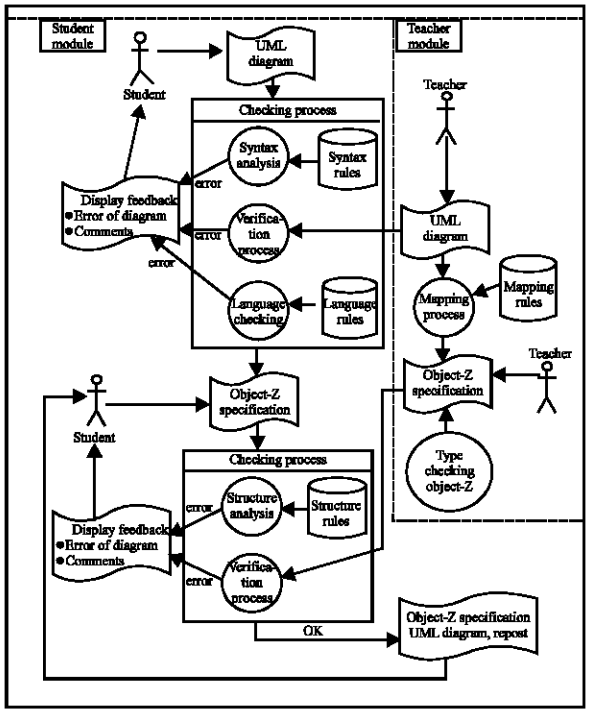

Fig. 5: Arichitecture of OOMA

- Verification Process
  This process will check whether the inputs of UML class diagram are accurate. This process will compare students' input with answer schemas by teachers.
- Language Checking.
  This process will check the naming of classes, attributes and operations. For example, for items in classes' name and its attributes, it is presume to use nouns as the naming while for its operations, it is presume to use verbs as the operations' name.

The above processes are carried out in order to verify that the input diagram is error free. Apart from that, it will also generate list of feedback to be used as guidance by the learners and hopefully it helps to improve their understanding in subject matter. On the other hand, the second input requires the students to provide the Object-Z specification. However, this input only available when the earlier received (i.e., during first input) class diagram is error free. The automatic checking process was done by the system is accomplished by comparing the students' diagram with the solution diagram, which was derived during the mapping process carried out within the teacher module.

The automatic checking process involves two activities namely the structure analysis and verification process. These activities are carried out to verify that the Object-Z specification is correct. As a result, a list of feedback will be generated from the two activities above (i.e., structure analysis and verification process). This feedback can be used as guidance for students to recognize the mistakes made by them during the Object-z specification design. In addition, the system also produces reports that provide details regarding to the students level of understanding and their level of basic knowledge in UML and Object-Z modeling.

CONCLUSION

In this study, we have presented the architecture of Object-Oriented Model Assessor (OOMA). Generally, the purpose of OOMA is to map UML class diagram model to Object-Z specification and to evaluate the input UML class diagram by the users. Although there are some tools developed for the same purpose, most of the tools implemented the process of mapping the Object-Oriented Model to formal specification and the mapping implementation is automatically done without explanation for the users. It makes the tools suitable for engineers or programmers but not suitable for users in academic fields. The suggested process in this prototypes are hoped to help students to have better understanding in the basic concepts of Object-oriented Model, UML and formal specification, Object-Z and finally UML class diagram. The objective of this research is to improve students' understanding on how to represent the system requirement in UML model and Object-Z specification model. In addition, it is to improve students' understanding on the relationship between these two models

REFERENCES

1. Smith, G., 2000. The Object-Z Specification Language, Advances in Formal Methods, Kluwer Academic Press Publishers.
2. Terwilleger, R.B., M.J. Maybee and L.J. Osterweil, 1989. An Example of Formal Specification as an Aid to Design and Development. Proceedings of The ACM SIGSOFT International Workshop on Formal Methods in Software Development.
3. Rumbaugh, J., M. Blaha, F. Premerlani and W.I. Eddy, 1991. Object-oriented Modeling and Design, London, Prentice Hall.
4. Johnston, W., 1996. A Type-Checker for Object-Z, SVRC Technical Report TR96-24, University of Queensland.
5. Hoggarth, G. and M. Lockyer, 1998. An Automated Student Diagram Assessment System. Proceedings of the 6th Annual conference on Integrating Technology into CSE-ITiCSE. Dublin City.

6. Durr, E.H. and J.V. Katwijk, 1992. VDM++ - A Formal specification language for object-oriented design, proceedings of the CompEURO IEEE Computer Society Press, pp: 214-219.
7. Wang, E., H. Richter and B. Chen, 1997. Formalizing and Integrating the Dynamic Model with OMT, Proceedings of the 19th SoftwareEngineering International Conference, pp: 45-55.
8. Weber, M., 1996. Combining StateCharts and Z for the Design of Safety-Critical Control Systems, FME'96:Industrial Benefit and Advances in Formal Methods, LNCS 1051, pp: 307- 326.
9. Meyer, E. and J. Souquieres, 1999. A Systematic Approach to Transform OMT Diagrams to a B Specification, FM'99, LNCS 1708, pp: 875-895.
10. Bertino, E., D. Castelli and F. Vitale, 1996. A Formal Representation for State Diagrams in the OMT Methodology, Proceedings of the SOFSEM : Theory and Practice of Informatics, 1175, pp: 328-334.
11. Dupuy, S., Y. Ledru and M. Chabre-Peccoud, 1997. Integrating OMT and Objek-Z, Proceedings of the BCS FACS/EROS.
12. Dupuy, Y.L. and M. Chabre-Peccoud, 1998. Translating the OMT Dynamic Model into Objek-Z, Proc. of the ZUM, pp: 347-366.
13. France, R.B., J.M. Bruel and M.M. Larrondo-Petrie, 1997. An integrated object-oriented and formal modeling environment, Object-Oriented Programming Journal.
14. Bruel, J.M. and R.B. France, 2000. Transforming UML Models to Formal Specifications, Proceedings of OOPSLA.
15. Kim, S.K. and D. Carrington, 2000. A Formal Mapping between UML Models and Object-Z Specifications, Proceedings of the ZB2000, pp: 2-21.
16. Kim, S.K. and D. Carrington, 1999. Formalizing the UML class diagram using Objek-Z, Proceedings of the 2nd IEEE conference on UML, pp: 83-98.